React ja relaatiotietokannat

> Opintojakson viimeisessä luvussa luodaan Reactilla käyttöliittymä sovellukseen, jonka dataa haetaan ja muokataan MySQL-relaatiotietokannasta AJAXia käyttäen PHP:n tarjoamista JSON-muotoisista rajapinnoista. Tietokannan ja PHP:n saattaminen toimintakuntoiseksi saattaa olla työlästä aiemmin näihin asioihin tutustumattomalle. Luvussa esitetään vaiheet sovelluksen käyttöönottamiseksi, mutta React-toiminnallisuuden opiskelu lähdekoodeja tutkimalla on jätetty tässä materiaalissa oman aktiivisuuden varaan. Myös PHP:n ja MySQL:n käyttöönoton opiskelu on jätetty osin oman aktiivisuuden varaan.

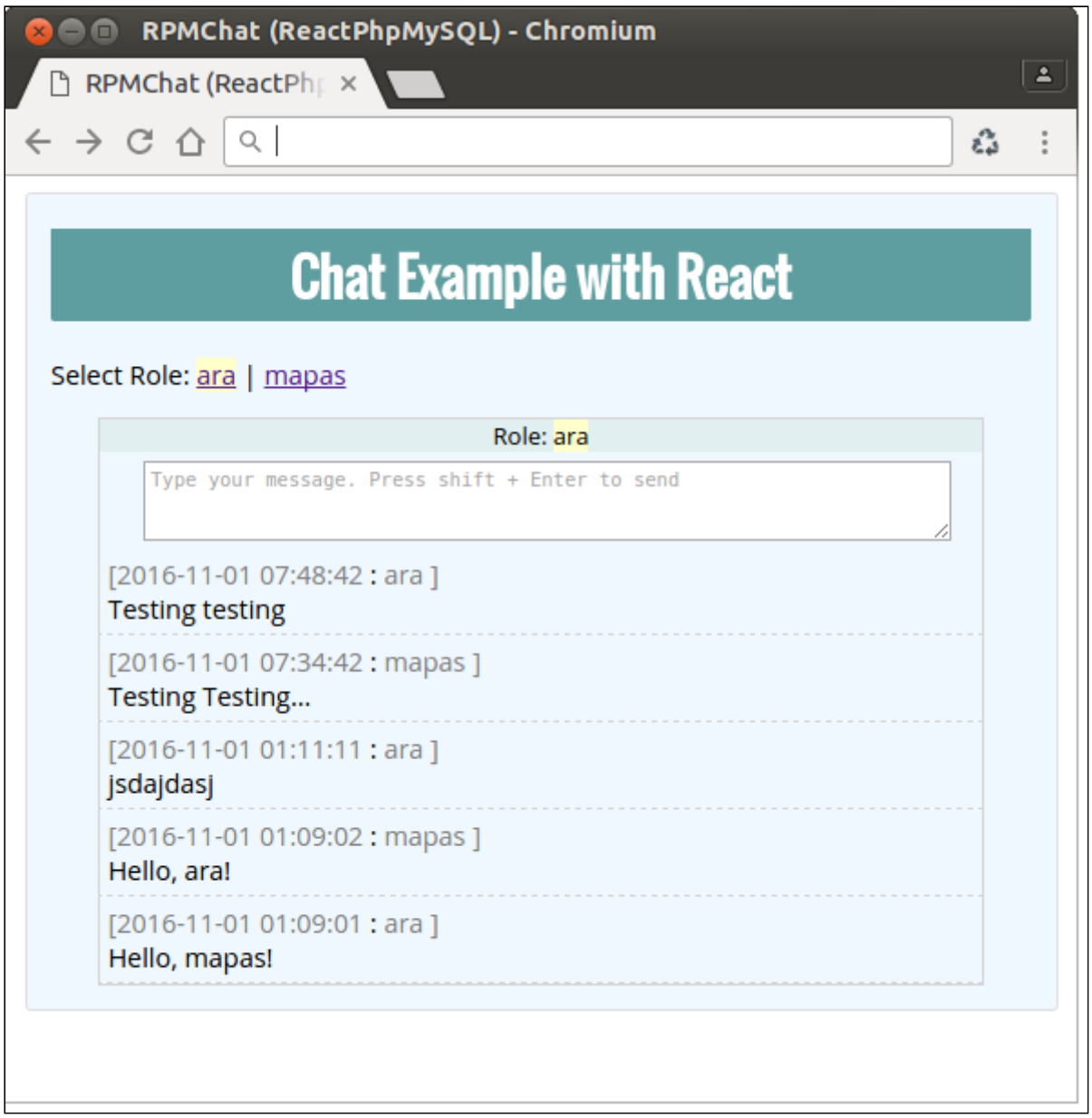
OnLine-Chat-sovellus

Esitetään React-kirjaston avulla toteutettu OnLine-Chat-sovellus, jossa käyttäjät ja heidän lähettämänsä viestit on tallennettu MySQL-relaatiotietokantaan. Toteutettavassa versiossa tietokantaan lisätään vain kaksi käyttäjää (=käyttäjäroolia) ja käyttäjien hallinnoimiseksi ei tarvitse olla olemassa käyttöliittymää.

Sovelluksen käyttäjä voi valita käyttäjärooliin (kuka kirjoittaa) autentikoitumatta sovellukseen. Kun käyttäjä lähettää valitsemassaan roolissa viestin, se tallennetaan MySQL-tietokantaan käyttämällä PHP-ohjelmaa, jota kutsutaan React-komponentista käyttämällä Jquery-kirjaston Ajax-toiminnallisuutta. Viestilista päivitetään käyttöliittymässä Reactin avulla vähintään 5 sekunnin välein uudelleen.

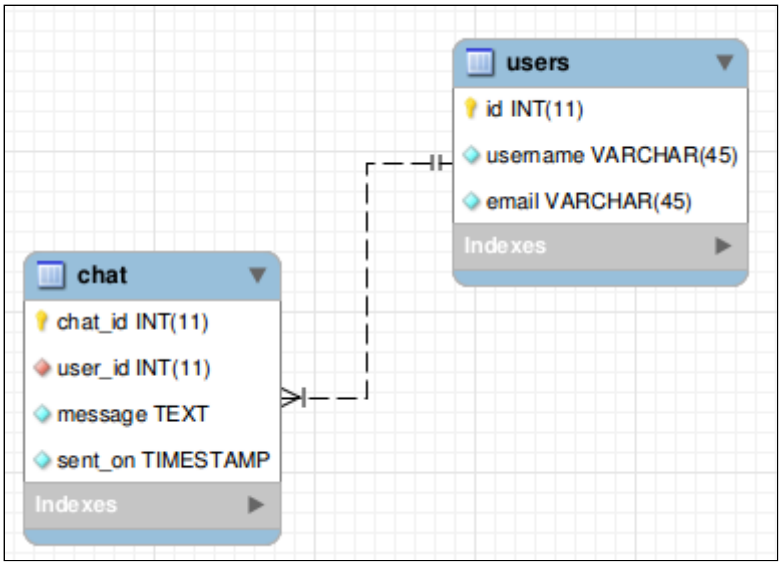
Tallennetut viestit luetaan MySQL-tietokannasta käyttämällä PHP-ohjelmaa, jota kutsutaan React-komponentista käyttämällä Jquery-kirjaston Ajax-toiminnallisuutta. Sovelluksen käyttöliittymä on oheinen, josta on pääteltävissä muu tarvittu toiminnallisuus.

Ohjelman käyttöliittymä on oheinen



1. Luodaan tietokanta

Ohjelmaa varten luodaan oheista ER-kaaviota noudattava MySQL-tietokanta



Ilman MySQL WorkBench-ohjelmistoa tietokannan voi alustaa seuraavilla SQL-lauseilla

```
  DROP TABLE IF EXISTS `chat`;
DROP TABLE IF EXISTS `users`;

CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `chat` (
  `chat_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `message` TEXT NOT NULL,
  `sent_on` TIMESTAMP NOT NULL,
  PRIMARY KEY (`chat_id`),
  KEY `chat_userid_fk` (`user_id`),
  CONSTRAINT `chat_userid_fk` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


INSERT INTO users (username, email)  VALUES ('ara', 'Ari.Rantala@jamk.fi');
INSERT INTO users (username, email)  VALUES ('mapas', 'Pasi Manninen@jamk.fi');

INSERT INTO chat (user_id, message, sent_on)  VALUES (1, 'Hello, mapas!', now());
INSERT INTO chat (user_id, message, sent_on)  VALUES (2, 'Hello, ara! ', now());
```

Jos testaat tätä sovellusta Student-palvelimella, käytössäsi on seuraavat ohjeet. Kun olet lisännyt tietokantaan yllä kuvastusti pari käyttäjää ja pari testiviestiä, testaa SQL-lausein että data todella on tietokannassa:

```
  SELECT * FROM users;
  SELECT * FROM chat;
```

2. Luodaan sovelluksen kansiorakenne ja tiedostot

- Kansiot ovat oheisessa listauksessa ilman tiedostotarkenninta. LabraUID tarkoittaa labraverkon käyttäjätunnustasi.
- Huomaa, että RpmChatDb.php **EI** ole tallennettuna public_html-kansion alle.

```
/
└──home
    └── labraUID
        ├── public_html
        │   └── chat
        │       ├── ajax
        │       │   ├── add_msg.php
        │       │   ├── get_all_users.php
        │       │   ├── get_messages.php
        │       │   └── get_user.php
        │       ├── js
        │       │   └── react-chat.js
        │       ├── react-chat-index.php
```

```
        ├── rpm-chat-config.php
        └── style
            └── core.css
    └── rpmchat-core
        └── RpmChatDb.php
```

- Kaikki yllä kuvatut tiedostot ovat saatavilla pakattuna tässä paketissa

  Tiedostoja täytyy osin muuttaa vastaamaan omia asetuksiasi. Tarvittavat muutokset ovat:

  1. `rpm-chat-config.php`-tiedoston rivi

  ```
  define ('__DBCONFIG_PATH', '/home/ara/rpmchat-core');
  ```

  tulee muuttaa vastaamaan omaa asetustasi eli Student-palvelimella sinun tulee muuttaa vain kohta **ara** vastaamaan omaa käyttäjätunnustasi

  2. `/home/SINUNTUNNUS/rpmchat-core`-kansion `RpmChatDb.php`-tiedoston tietokanta-asetukset tulee asettaa vastaamaan omia asetuksiasi riveillä 9 ja 10:

  ```
    ('mysql:host=mysql.labranet.jamk.fi;dbname=SINUNTIETOKANNANNIMI;charset=ut
                              'SINUNLABRATUNNUS', 'SINUNMYSQLSALASANA');
  ```

3. Testaa sovellusta

Sovelluksen toiminnan testaaminen kannattaa aloittaa kutsumalla web-selaimen kautta ajax-kansion get_all_users.php-skriptiä. Riippuen sovelluksesi asennuskansiosta osoite voi olla esim. http://student.labranet.jamk.fi/~SINUNTUNNUS/chat-test/ajax/get_all_users.php

Ohjelman pitäisi tulostaa käyttäjälista JSON-muodossa seuraavasti:

```
[{"user_id":1,"name":"ara","email":"Ari.Rantala@jamk.fi"},
{"user_id":2,"name":"mapas","email":"Pasi Manninen@jamk.fi"}]
```

Viimeisenä vaiheena on testata varsinaisat sovellusta **react-chat-index.php** kutsumalla sitä web-selaimen kautta. Riippuen sovelluksesi asennuskansiosta osoite voi olla esim. http://student.labranet.jamk.fi/~SINUNTUNNUS/chat-test/react-chat-index.php

Jätetty tarkoituksella tyhjäksi